Application of Hybrid Computing Technologies for High-Performance Distributed NFV Systems

Mikhail M. Rovnyagin[1], Alexey A. Kuznetsov[2]
Department of Computer Systems and Technologies
National Research Nuclear University MEPhI (Moscow Engineering Physics Institute)
Moscow, Russia
[1]m.rovnyagin.2015@ieee.org, [2]AAKuzneysov@mephi.ru

***Abstract* — Currently, the areas of transmission and distributed processing technology commonly used network functions virtualization (NFV). A key feature of these solutions is that they can use in the Cloud. Modern cloud infrastructure often has the GPGPU-coprocessors for acceleration purposes. There are a number of problems in which it is possible to use hybrid computing technology to speed up the NFV: encryption, pattern matching, media content processing (compression, audio/video processing, etc.). In this paper, we propose a new NFV-GPGPU architecture and provide the results of its study in case of AES network virtualized function acceleration.**

Keywords — High performance computing, NVIDIA CUDA, Encryption, hybrid architecture, NFV, OpenCL

I. INTRODUCTION

Increasing the number of users who operate network interaction for different purposes (social networks, content view, data sharing, etc.) is one of the main issues in the field of telecommunication technologies. Due to the growth of data processing volumes, network providers are not able to maintain the existing architecture because of high investing costs.

In 2012, the European Telecommunications Standards Institute offered a concept of network functions virtualization – a technology allowing the network providers to improve the quality of the services they are offering, optimize hardware costs, flexibly manage the existing and new services.

The network functions virtualization technology is peculiar in a way that specialized network devices are replaced with software ones, which do not depend on hardware implementation. For example, a network resource administrator does not need to have their own router, because this router can be software-based on a virtual machine on the network provider's end.

Contemporary cloud services already have GPGPU co-processors for accelerative purposes, such as calculation of hash tables, or hash functions in the Bloom filter [1]. On the other hand, virtualization of network functions has a big number of tasks, which require increasing the speed of network function virtualization (NFV).

Based on this, we are offering a new approach towards network function virtualization using GPGPU co-processors (NFV-GPGPU). This approach combines the benefits of both NFV technology and GPGPU architecture, using which one can achieve a considerable performance increase. The only limit is the ability of function to paralleling.

To present the results, we offer one of such network functions with the hybrid computing model. Since network interaction often utilizes encryption, we took the standard encryption algorithm of AES (Advanced Encryption Standard). The parallel version of this algorithm is implemented using OpenCL technology and presented in the form of NFV virtual network function.

978-1-5090-4865-6/17/$31.00 ©2017 IEEE

II. RELATED WORKS

Study on paralleling a separate VNF function on a graphic processor was conducted by the authors in the paper [2]. The study discusses paralleling of chunk classification function used in the Internet routers, which resulted in achieving acceleration by 1.9 times.

In the paper [3], the authors offered to parallel a match table in OpenFlow multiplexer, as the match table is a critical function there. The Gflow algorithm has reached the speed by 8-10 times bigger than the solutions based on central processor computing.

In the article [4], the authors present a parallel version of encryption algorithm, based on hybrid real-time computing. The researchers have used CTR encryption function and GCM modes of AES128 algorithm used in PON systems. GPU-based algorithm has shown a considerable advantage in terms of performance over the bit-serial version.

The research in the field of parallel implementation of VNF function is also being conducted for multimedia applications. Thus, the authors of the article [5] have presented an encryption paralleling algorithm for high definition video.

In the paper [6], the authors present a service-oriented approach, based on arranging dynamic chains of virtual network functions (VNF) through software-defined networks. This approach includes all the advantages of not only virtual network functions, but also that of software-defined SDN networks, as well as service-oriented architecture, and allows arranging flexible VNF chains.

Our paralleling method is based on the article [7], where authors examine the methodology of block encryption algorithm paralleling and conclude that granular approach, which sends a data portion of 16 bytes to the flow, has the highest performance. The hybrid cluster has been built based on the principle described in the paper [8].

III. GENERAL ARCHITECTURE

We suggest using a network with the following VNF architecture within the described approach: network service provider, network clients, orchestrator, and virtual server. Network clients communicate with outer world through the provider, while all the data transfer traffic should be encrypted or decrypted using the AES encryption algorithm, which is installed as a VNF.

To build a chain of virtual network functions with the data encryption function using AES algorithm, we used the approach described in the papers [9,10].

All computing function of network architecture are transferred from hardware level to software. A server with a set of virtual machines having virtual network functions installed on them is used as the main device. The KVM visualization system allows organizing disjoint interaction: 1 virtual machine – 1 CUDA accelerator.

Currently, the concept of hybrid computing architecture (HSA, Hybrid System Architecture) has spread widely. This architecture enables conducting calculations using the CPU and GPU concurrently. For the interaction with hybrid architecture we use the OpenCL technology, which was developed by the manufacturers of both graphic and central processing units. This technology enables the user to choose the option of task completion individually based on the description of the hardware complex. The OpenCL software model is presented in Figure 2. We use 2 devices of a graphic accelerator with Nvidia Tesla processor and Intel Xeon central processor as Compute Devices. Each processor has its own Compute Units. For Nvidia Tesla, it is 13 SMX blocks, each of which utilizes 192 processing elements – CUDA Core. For Intel Xenon, the processing takes place at 8 cores (Compute Unit).

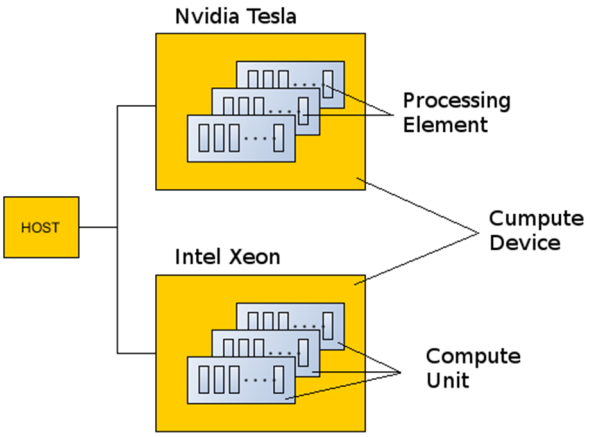

Fig. 2 OpenCL architecture model

We have envisaged three work options within the scope of the suggested NFV-GPGPU approach:

1) Running the computing only at the central processing unit (CPU-only).

2) Running the computing only at the graphics processing unit (GPU-only).

3) Automatic selection between GPU and CPU (Auto).

The following should be viewed as suitable for NFV technology acceleration: NVIDIA CUDA, Intel MIC, AMD Fire Stream. One should also consider that the majority of modern AMD processors support the HSA (Heterogeneous Systen Architecture) technology, which allows decreasing the overhead costs of data copying from CPU memory to GPU memory. Two approaches towards the implementation of VNF on hybrid computing hardware are possible: direct decompression of VNF modules and virtual machine decompression with the further *forwarding* of hybrid accelerators into them (Figure 3). The second option is intended for the use on powerful multi-processor servers, equipped with discontinuous co-processor. The first one is more suitable for the fully distributed systems, IoT and Fog Computing range technologies.

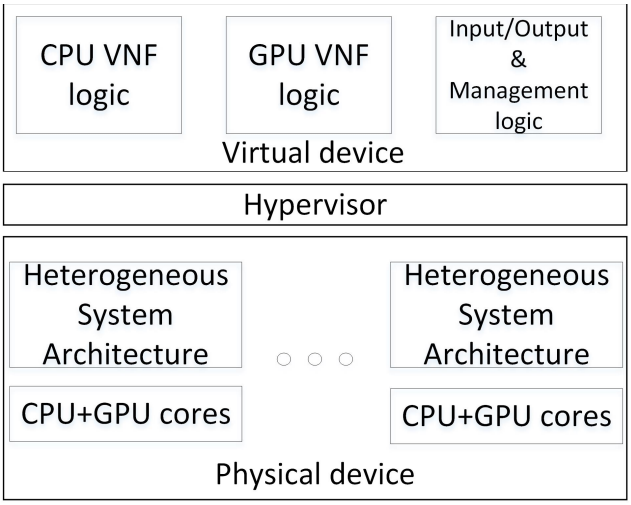

Fig. 3 Hybrid accelerator forwarding scheme

Forwarding an accelerator takes place through a freely developed Qemu-KVM hypervisor [11]. It can be installed on any modern Linux distribution.

IV. AES Encryption Algorithm

1) Sequential AES Algorithm

AES encryption algorithm [12] is an algorithm of symmetric block cipher, which can process the data blocks with 128-bit length using a 128-, 192-, and 256-bit length cipher. Length of S Input String: length(S) = 128 bits; Cipher text with length(C) = 128 | 192 | 256 bits. All bits are transformed to bytes.

AES algorithm is based on a 2-dimensional array called «current encryption state». It contains 4 strings of bytes, each of which contains Nb bytes, where Nb has 2 indicators.

AES algorithm includes the following main functions:

1) AddRoundKey() – transformation during encryption and reverse encryption, where the round key is added to the temporary encryption state using XOR.

2) MixColumns() – transformation during encryption that takes all the columns of the temporary encryption state and mixes their data (separately from each other)

3) ShiftRows() – transformation during encryption that processes temporary encryption state by repeatedly

shifting the last 3 strings of temporary encryption with different flush.

4) SubBytes() – transformation during encryption that processes the temporary encryption result using a non-linear byte override table (S-box), applying it at each byte state independently.

2) Parallel AES Algorithm

There are several ways of paralleling AES encryption algorithm. As it was mentioned before, the most efficient paralleling method is dividing 16-byte granular blocks into threads, which go to the graphic co-processor in a shape of blocks to the shared graphic memory, where they are run simultaneously.

However, the following paralleling options are possible within the scope of this concept: location of round keys, current state cache, location of replacement unit, and whether it is worth paralleling the totaling operation with an AddRoundKey() key in a different way.

To decrease the size of allocated memory during the transfer of parameters to the core function, we decided to load Sbox, reverse Sbox and Mix mixing matrix into the constant memory of the graphic processor. Considering the fact that the data processing blocks have quite a small size (16 bytes), the feasibility of AddRoundKey() function paralleling is no longer valid, as the overhead costs of memory allocation will considerably exceed the benefit from parallel processing.

In our variant of AES encryption algorithm paralleling, all the input data are processed simultaneously, except for the case when the graphic processor cannot contain such large data volume at a time. In that case, we will have to divide the input data into parallel processing groups.

The first stage prepares the graphic processor for data processing, initializes the cores and creates write buffers. The second stage loads the data into the graphic processor's memory. The third stage performs parallel computing of 16-byte initial data groups.

Then, all the groups are loaded out of the GPGPU memory and transformed into the input data. Figure 4 shows the main idea of encryption algorithm paralleling.

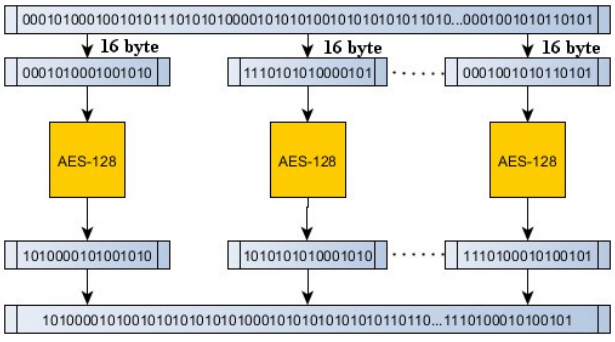

Fig. 4 AES paralleling algorithm

Comparing to the bit-serial version of the algorithm, the parallel version works a few times faster. Work time comparison graph for bit-serial and parallel algorithm is presented in Fig. 5.

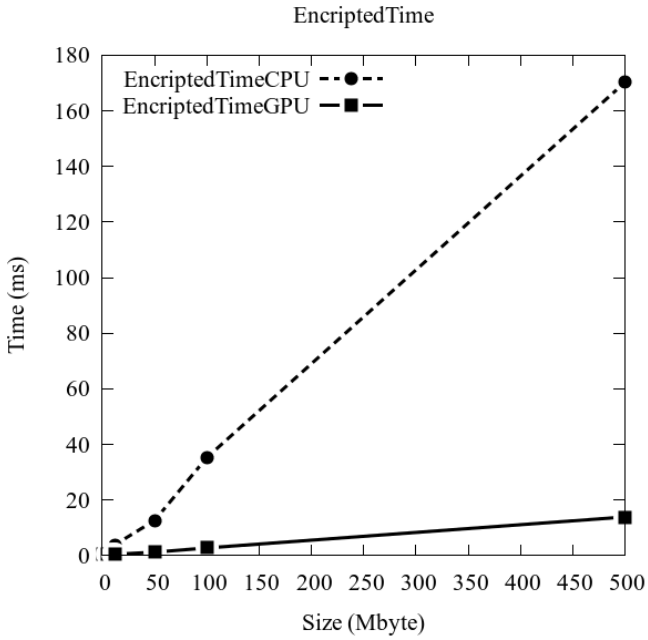

Fig. 5 Work time of single flow and GPU version of AES algorithm

V. Flow and Chunk Transmission

In our work, we based on the research [7], assuming that virtual network functions may be represented as "chains".

To transfer data between the chain sections to the virtual AES encryption function, we used 2 methods of network transmission: the first one is streaming mode, the second one is chunked mode.

It was done to achieve more flexibility in the implemented system. VNF-AES encryption system is implemented using the OpenCL technology and may be unwrapped both on servers having a graphical component and servers without one.

As it is known, central and graphic processors have different specific nature when building a network interaction. For the central processor, distinctive is processing of stream information when data come consecutively, one after one. A completely different approach is specific for graphic processor. As there is a bigger number of cores in the graphic processor (currently it can reach a few thousand), initiation of all cores of the processor will be the most efficient way of data processing.

That is why the streaming mode was chosen for the central processor, and the chunk mode for the graphic processor respectively, which indicates the fact that the central processor will process the data as it comes in, while the graphic processor will accumulate a specific amount of data before starting to process them.

VI. Implementation of Stream and Chunk VNF Transfer

Data transmission between the sections of a virtual network function chain was implemented in the C programming language, with a set of software libraries of Berkeley sockets <sys/socket.h>, <arpa/inet.h>.

For the experiment, we wrote conditioned prototypes of "Server" and "Client" sharing files of predefined size with each other. Each experiment was conducted 100 times to average the experimentally obtained data.

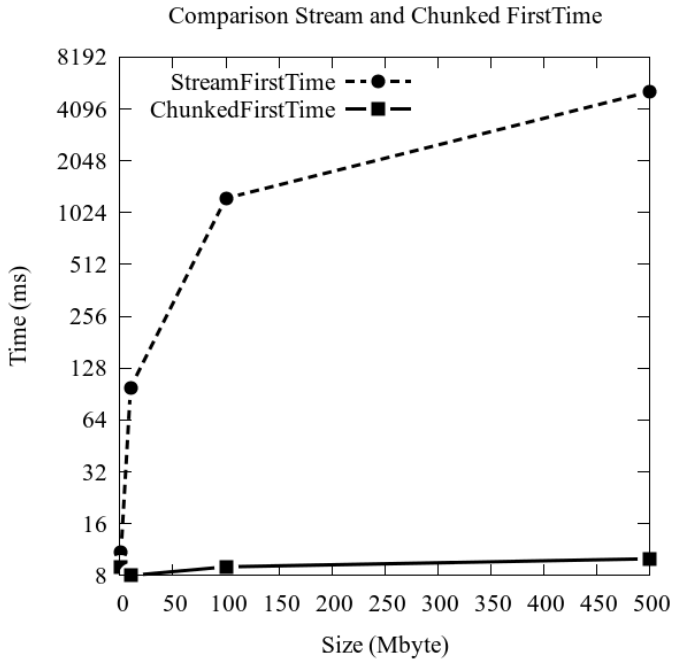

Fig. 6 Comparison of first element access speed

As you can see from figure 6, the stream transmission with increasing the volume of transferred data starts spending more memory, placing the transmitted data into the random-access memory, and creating a swap file when there is not enough memory. Also, when transmitting data in the streaming mode, the access time to the first data is increased depending on the amount of the transmitted data.

A completely different situation is observed in the chunk mode. In case of increase in the amount of transmitted data, the access time to the first chunk does not depend from the amount of transmitted data in any way, and the memory spent for accepting the transmitted data remains constant during the whole transmission.

Conclusion

Results presented show that the suggested new NFV-GPGPU architecture allows a considerable increase in the productivity of such network function as AES algorithm encryption. Based on the results, we can see that the acceleration of the algorithm on a graphic processor gives the acceleration by dozens of times comparing to the single-flow version. At the same time, with small data volumes, when the efficiency of GPU-acceleration drops, the CPU implementation in the current mode of network interaction starts working.

This architecture allows accelerating not only VNF AES, but also a range of other tasks, for example: video/audio stream encryption, compression, traffic routing.

References

[1] A. A. Dyumin, A. A. Kuznetsov, and M. M. Rovnyagin, “Evaluation of statistical properties of a modified Bloom filter for heterogeneous GPGPU-systems,” in 2015 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW), 2015, pp. 71–74.

[2] S. Zhou, S. G. Singapura, and V. K. Prasanna, “High-performance packet classification on GPU,” *2014 IEEE High Perform. Extrem. Comput. Conf. HPEC 2014*, 2014.

[3] K. Qiu, Z. Chen, Y. Chen, J. Zhao, and X. Wang, “GFlow: Towards GPU-based high-performance table matching in OpenFlow switches,” in International Conference on Information Networking, 2015, vol. 2015–Janua, pp. 283–288.

[4] A. O. and T. H. Takahiro Suzuki, Sang-Yuep Kim, Jun-ichi Kani, Ken-Ichi Suzuki, “Parallelization of Chipher Algorithm on CPU/GPU for Real-time Software-Defined Access Network,” in APSIPA Annual Summit and Conference 2015, 2015, no. December, pp. 484–487.

[5] Comi et al., “Hardware-accelerated high-resolution video coding in Virtual Network Functions,” 2016, pp. 32–36.

[6] B.Martini, F.Paganelli “A Service-Oriented Approach for Dynamic Chaining of Virtual Network Functions over Multi-Provider Software-Defined Networks” National Interuniversity Consortium for Telecomunications (CNIT), Febrary 2016. (*references*)

[7] K. Iwai, N. Nishikawa, and T. Kurokawa, “Acceleration of AES encryption on CUDA GPU,” *Int. J. Netw. ...*, vol. 2, no. 1, pp. 131–145, 2012.

[8] N. P. Vasilyev and M. M. Rovnyagin, “Hybrid Clusters for Budget Supercomputers and Cloud Computing,” Autom. Remote Control, vol. 75, no. 10, pp. 1869–1874, Oct. 2014.

[9] I. V. Chugunkov et al., “Three-dimensional data stochastic transformation algorithms for hybrid supercomputer implementation,” in Proceedings of the Mediterranean Electrotechnical Conference - MELECON, 2014, pp. 451–457.

[10] I. V. Chugunkov, O. Y. Novikova, V. A. Perevozchikov and S. S. Troitskiy, "The development and researching of lightweight pseudorandom number generators," *2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW)*, St. Petersburg, 2016, pp. 185-189.

[11] Qemu-KVM: work in Debian (2013). Available at https://habrahabr.ru/post/170259/ (accessed 20 November 2016)

[12] Advanced Encription Standart (AES), Federal Information Processing Standards Publication 197, 21 nov. 2001